Compare and contrast various software development methodologies, such as Agile, Scrum, and DevOps, discussing their advantages, challenges, and best practices

check for updates

Article history:
Received:
April/12/2020
Accepted:
Nov/08/2020

Ranadeep Reddy Palle
Software engineer and independent researcher

Abstract

The study presents the different Agile, Scrum, and DevOps approaches in software development. These approaches are compared to identify their strengths, and weaknesses as well as the practical ways of using them. Adopting the approach of comparative analysis, it examines management principles, efficiency, adaptability, and productivity characteristics. This research assumes methodological integration of literature and case study approaches to chart project success with consideration of organizational culture. Agile highlights agility while Scrum provides structured iterations, and DevOps acts as a tool to integrate the development and operations. The conclusion mandated the adaptation of the methodology option to the project requirements and corporate culture for the purpose of greater efficiency, productivity and product quality.

Keywords: *Agile, Scrum, DevOps, software development methodologies, comparative analysis, efficiency, adaptability, productivity.*

Introduction

Software development methodologies are key in the outcomes of software projects in turn determining project success by virtue of the influence that they have on team dynamics. Agile, Scrum, and DevOps have become prominent methods that provide different routes for building software applications. Agile is focused on adaptability and customer collaboration, Scrum offers iterative development through well-planned-out sprints, and DevOps bridges development and operations for continuous delivery. Getting to know the core ideas, advantages, and challenges of the methodologies is essential for the current software teams and companies that want to improve the performance and quality of their products. This study focuses on Agile, Scrum, and DevOps methodologies but compares The implication or best practice with software development contexts.

Objectives

RO1: To understand the underpinning principles and methodologies of Agile, Scrum, and DevOps in order to know their distinctive approaches to software development.
RO2: To discuss the advantages and benefits offered by Agile, Scrum, and DevOps methodologies from the perspective of efficiency, adaptability, and productivity.

RO3: To evaluate the challenges and limitations of each methodology including the possible implementation roadblocks and scale-up issues.
RO4: To investigate best procedures and tactics for efficient implementation and assimilation of Agile, Scrum, and DevOps methodologies within software development teams and organizations

Methodology

The study uses a comparative analysis method to contrast Agile, Scrum, and DevOps approaches within software development. Through the review of already published literature and case studies, the core themes, advantages, disadvantages, and best practices associated with each approach have been discovered. The study emphasizes effectiveness, adaptability, and efficiency areas, analyzing implementation experiences in various circumstances. Through the combination of evidence from different studies, the aim should be the formation of an overall picture of the special methods of Agile, Scrum, and DevOps. This methodology aids in spotting the crucial variables that influence software development method selection, project success, and organizational culture, and helps in making more informed decisions for stakeholders while executing software development projects.

Introduction to Software Development Methodologies

Software development methodologies are structured approaches that provide directions on planning, development and management of software projects. The selection of the suitable methodology is a key factor because it determines the success of the project, it influences the development processes, the team dynamics, and the outcomes. With the changing dynamics, Agile, Scrum, and DevOps have taken over as the most used methodologies, each of which has a unique way of developing software [1]. Agile advocates adaptability and collaboration with the customer, iterating the development process to accommodate changing needs. Scrum, in the Agile paradigm, builds the teams around brief, concentrated sprints, which in turn promote transparent and accountable systems. DevOps assembles development and operation and encourages collaboration and automation to smoothen the way of software delivery and deployment processes. An awareness of these methodologies is crucial for software teams and organizations that aim to boost efficiency, productivity, and product quality [2]. This introductory part paves the way for a more detailed discussion of Agile, Scrum, DevOps, and similar approaches, which are of great importance in modern software development.

Agile Methodology

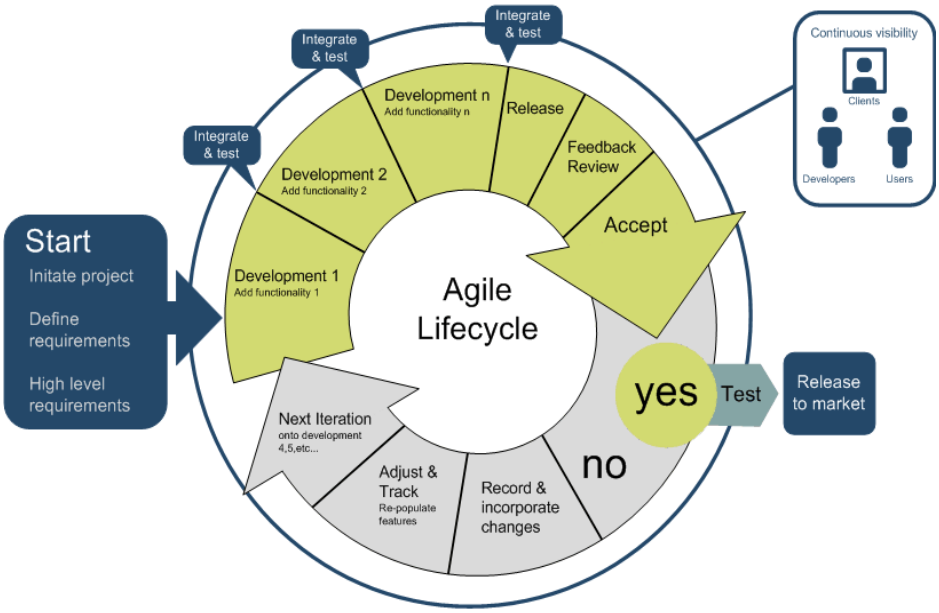

Figure 1: ***Agile Methodology in Software Development***
(Source: [13])

Principles and Philosophy

Agile methodology is based on the Agile Manifesto, which stresses the individuals and interactions, working software and customer collaboration over the processes and tools in the first place. The fundamental Agile principles are those of - meeting customer needs through early and regular delivery of valuable software, accepting changing customer expectations, delivering working software often, and basing projects on individuals who are self-driven [1].

Advantages

Flexibility and Adaptability: Agile methodology provides an agile and adaptable process of development which can enable teams to rapidly respond to the dynamic changes in the requirements and in the market.
Customer Satisfaction: Agile methodology promotes the involvement of the customer in every development phase and multiple deliveries of workable software to ensure high customer satisfaction [3].
Continuous Improvement: One of the core principles of Agile is continuous improvement driven by iterative development cycles when teams ensure the feedback is incorporated and enhancements are made throughout the project lifecycle [3].

Challenges

Resistance to Change: Adopting Agile methodology may meet opposition from the team members who have been practicing the traditional development approaches, so an inner culture modification will be needed.

Lack of Documentation: Agile's emphasis on working software as opposed to heavy documentation can cause difficulty in information and knowledge transfer which in turn could create barriers to future maintenance.
Scope Creep: While Agile has an iterative approach and a focus on responding to changes, project requirements may easily go out of control if they are not properly managed and prioritized [4].

Best Practices

Iterative Development: Agile promotes iterative development cycles, which enable teams to provide the list value with each iteration and collect feedback for periodical improvement.
Collaboration and Communication: Agile is all about collaboration, communication, and transparency among team members, stakeholders as well as customers to ensure teams are fully aligned and a perfect understanding is achieved [5].
Retrospectives and Feedback Loops: The recurrent retrospectives and feedback fit is a feel where teams reflect on their process, spot causes of slack, and implement the revisions to enhance their performance and productivity.

Scrum Methodology

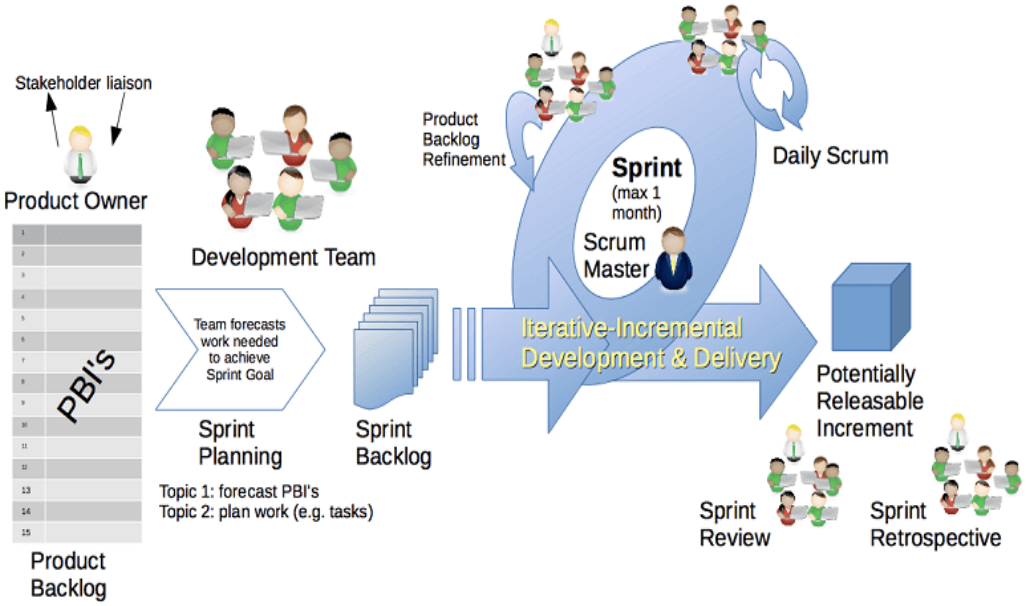

Figure 2: ***Scrum methodology in Software Development***
(Source: [14])

Framework and Roles

Scrum is an Agile framework that was created for solving complex adaptive problems in terms of the productivity and creativity of the development function. It includes an iterative process and self-organization and simplicity are the two principles it focuses on. Within

the Scrum framework, there are key roles: Product Owner, Scrum Master, and Dev Team [6]. The Product Owner is the link between the stakeholders and his role is to maximize the value of the product. The Scrum Master takes the role of facilitator of the Scrum process and helps the team to follow Scrum principles and practices. It is the Development Team that is supposed to build and deliver the increments of a functional product at the end of every Sprint.

Advantages

Transparency and Accountability: Scrum fosters transparency by showcasing movement in the execution of tasks using artifacts like the Product Backlog and Sprint Backlog. It establishes responsibility and accountability among the team members by delineating the roles and responsibilities.
Predictability: Using defined Sprints of standard duration and also the Sprint Reviews, Scrum has a steady pace of product development, facilitating stakeholders to foresee delivery times and plan accordingly [7].
Empowered Teams: Scrum enables multifunctional teams to coordinate themselves and make decisions jointly, which results in increased ownership, motivation and innovation.

Challenges

Timeboxing Issues: It can be tricky to find the appropriate balance of timeboxing and flexibility because very strict time frames could cause incomplete or rushed work and overly flexible timelines risk to have scope creep.
Overemphasis on Meetings: Scrum places significant demands on the daily Scrum meetings and frequent reviews which can consume a lot of resources and may be regarded as time-consuming by a part of team members [7].
Team Dynamics: Teams must be good at coordinating and communicating in Scrum, and team issues like conflicts or division can keep the team from moving forward or prevent it from achieving the desired outcomes.

Best Practices

Sprints and Backlogs: Sprints consist of time-boxed iterations, during which the Development Team releases possibly shippable portions of the product. The Product Backlog is a prioritized list of features and requirements, while the Sprint Backlog shows the tasks that are to be accomplished during the Sprint [8].
Daily Stand-ups: Daily Scrums, or stand-ups, are a chance for the team to synchronize their efforts, talk about the progress and figure out any roadblocks or impediments.
Product Owner Engagement: An active partnership with the Product Owner is crucial for enabling the team to comprehend and implement the value that is aligned with the stakeholder needs and priorities [8].

DevOps Methodology

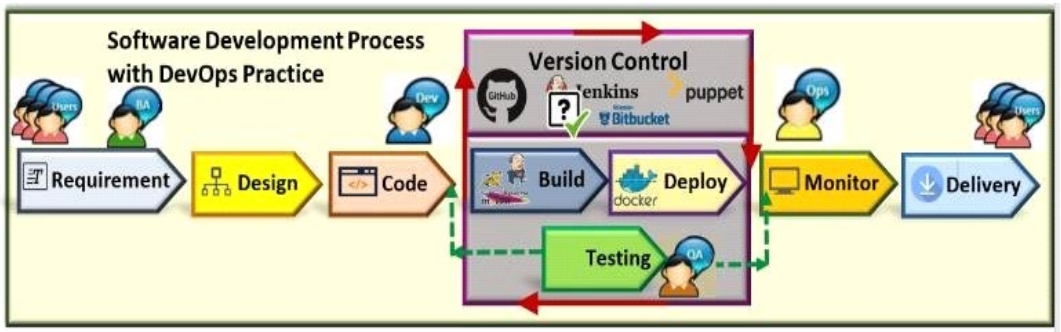

Figure 3: ***DevOps methodology in Software Development***
(Source: [15])

Integration of Development and Operations

DevOps is a methodology that combines development (Dev) and operational (Ops) teams to optimize the software development life cycle. It bursts former walls between these two functions, thus promoting collaboration and communication all the way from code development to deployment and maintenance [9].

Advantages

Continuous Delivery: DevOps makes possible continuous delivery of software updates which allows organizations to release new features and fix bugs quickly and regularly. Via automated pipelines, code changes are tested, integrated, and deployed in a one-motion operation that ensures the whole release process is smooth [10].
Automation and Efficiency: DevOps is centered on automation, which in turn speeds up the process of delivering software by automating mundane tasks like building, testing, and deployment. This lowers manual errors, shortens development cycles, and opens up resources for more valuable activities.
Improved Collaboration: DevOps thrives in an environment of collaboration and shared responsibility among development, operations, and relevant teams. Removing the barriers and bringing in cross-functional teamwork DevOps promotes better communication, quicker resolving of issues, and efficiency [11].

Challenges

Cultural Resistance: The process of DevOps implementation happens to be opposed by cultural barriers that exist within organizations. The use of traditional hierarchical structures and reluctance to change may complicate the adoption of DevOps practices, therefore, the cultural transformation and thinking patterns must be altered.
Toolchain Complexity: The vast repertoire of DevOps tools and technologies often leads to complexity and integration issues. Handling and coordination of different tools used in different stages of development and production environments is both an expertise and a careful orchestration activity [10].
Security Concerns: Security is often neglected in the midst of the fast-paced development and deployment in DevOps. Implementing strong security controls along the pipeline such

as code scanning, access control, and vulnerability management, would help in minimizing the risks.

Best Practices

Infrastructure as Code (IaC): IaC makes it possible to provision and manage the infrastructure by using code, making the solution consistent, and scalable as well as introducing the version controls. It is used for automated infrastructure deployment and configuration, thus avoiding errors made in a manual way and ensuring the reproducibility of outcomes [9].

Continuous Integration/Continuous Deployment (CI/CD): CI/CD practices of automation include integration of code changes, test runnings and deployment of applications. Through small and iterative releases that are done frequently and always work, CI/CD helps organizations become agile, decrease time to market, and improve the quality of software [12].

Monitoring and Feedback Mechanisms: Monitoring and feedback mechanisms should be robust to avoid deficiencies or discords in the DevOps model. Processing and infrastructure performance in real-time gives great information about problems and optimizing systems quality metrics. Feedback loops allow teams to reiterate the processes that continuously drive them towards the improvement or perfection of their systems.

Comparative Analysis

Agile, Scrum and DevOps share common features of the software development process and of project management. What is more, they are concerned with cooperation between all team members and stakeholders. This communication that is open and focused on teamwork, develops a space for the exchange of ideas and all the members being in alignment with the project objectives. In addition to that, Agile, Scrum, and DevOps promote iterative development processes allowing the teams to continuously deliver improvements with the help of the stakeholders' feedback. Using an agile approach, the team can easily react to changing requirements and market changes, thus, producing the product that meets the customer demand as closely as possible. In addition, all these methodologies heavily rely on the value delivery to customers through customer involvement in the approach of incremental delivery and feedback-driven development practice that ultimately gives high customer satisfaction and product success. In spite of the fact that they have many common aspects, Agile, Scrum, and DevOps still show their distinctive nature through their scope and focus. The Agile methodology supports rapid project management which concentrates on flexibility and adapting to changes. Contrary to popular belief, scrum is an iterative development methodology that provides a scalable and stable framework for the management of a project within the constraints of a certain time frame with subsequent iterations done in short, focused sprints of development. In addition to the development phase, DevOps can work throughout the whole software delivery lifecycle including the deployment and operations. It uses the holistic method to make the delivery process smooth, to cut the bottlenecks, and consequently to upgrade the overall organization. The scope or type of methodology is, therefore, an important factor in choosing the right one for a project that fits different organizational cultures. Agile is aligned with projects characterized by flexibility, with a high need for iterations, thus allowing the team to cope with the changing requirements and circumstances. On the

contrary, Scrum seems to be more fitting for projects with well-defined requirements and urgency to get feedback, it enables iterative development and stakeholder interaction via a structured framework. DevOps is excellent for those companies that step out for the purpose of optimizing the entire software delivery lifecycle thus creating a culture of collaboration, automation, and continuous improvement.

Conclusion

The comparative study has led to the discovery that Agile, Scrum and DevOps methodologies have common principles of collaboration, iterative development and customer focus, while they vary in scope and focus. Agile is about flexibility, Scrum is a more structured approach, and DevOps integrates development and operations within the software delivery cycle. This illustrates the fact that matching the methodology choice with project requirements and organizational culture should be the priority. Increased efficiency, productivity, and product quality can be gained by software development teams and organizations once methodologies' advantages, challenges, and best practices are understood. The focus of future research should be on deeper examination and optimization of the already existing approaches to ensure the efficiency of software development.

Bibliography

[1]. P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile Software Development Methods: Review and Analysis," *Arxiv.org*, 2017. https://arxiv.org/abs/1709.08439

[2]. T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213–1221, Jun. 2012, doi: https://doi.org/10.1016/j.jss.2012.02.033.

[3]. L. Vijayasarathy and D. Turk, "Drivers of agile software development use: Dialectic interplay between benefits and hindrances," *Information and Software Technology*, vol. 54, no. 2, pp. 137–148, Feb. 2012, doi: https://doi.org/10.1016/j.infsof.2011.08.003.

[4]. Y. Liang, "Analysis and algorithms for parametrization, optimization and customization of sled hockey equipment and other dynamical systems." 2020.

[5]. J. Cho and Joey, "An Exploratory Study on Issues and Challenges of Agile Software An Exploratory Study on Issues and Challenges of Agile Software Development with Scrum Development with Scrum Recommended Citation Recommended Citation," 2010. Available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1595&context=etd

[6]. P. Meso and R. Jain, "Agile Software Development: Adaptive Systems Principles and Best Practices," *Information Systems Management*, vol. 23, no. 3, pp. 19–30, Jun. 2006, doi: https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93704.3.

[7]. J. Noll, M. A. Razzak, J. M. Bass, and S. Beecham, "A Study of the Scrum Master's Role," *Product-Focused Software Process Improvement*, pp. 307–323, 2017, doi: https://doi.org/10.1007/978-3-319-69926-4_22.

[8]. Y. Liang, J. R. Alvarado, K. D. Iagnemma, and A. E. Hosoi, "Dynamic sealing using magnetorheological fluids," Physical Review Applied, vol. 10, no. 6, p. 64049, 2018.

[9]. Moniruzzaman, A B M and Hossain, "Comparative Study on Agile software development methodologies," *arXiv.org*, 2013. https://arxiv.org/abs/1307.3356

[10]. T. Stober and U. Hansmann, *Agile Software Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. doi: https://doi.org/10.1007/978-3-540-70832-2.

[11]. J. Hamunen, "Challenges in adopting a Devops approach to software development and operations," *aaltodoc.aalto.fi*, 2016, Available: https://aaltodoc.aalto.fi/handle/123456789/20766

[12]. L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, "DevOps Adoption Benefits and Challenges in Practice: A Case Study," *Product-Focused Software Process Improvement*, pp. 590–597, 2016, doi: https://doi.org/10.1007/978-3-319-49094-6_44.
[13]. L. BANICA, M. RADULESCU, D. ROSCA, and A. HAGIU, "Is DevOps another Project Management Methodology?," *Informatica Economica*, vol. 21, no. 3/2017, pp. 39–51, Sep. 2017, doi: https://doi.org/10.12948/issn14531305/21.3.2017.04.
[14]. R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps?," *Proceedings of the Scientific Workshop Proceedings of XP2016 on - XP '16 Workshops*, 2016, doi: https://doi.org/10.1145/2962695.2962707.
[15]. X. Wu, Z. Bai, J. Jia, and Y. Liang, "A Multi-Variate Triple-Regression Forecasting Algorithm for Long-Term Customized Allergy Season Prediction," arXiv preprint arXiv:2005.04557, 2020.
[16]. K. Beck *et al.*, "Manifesto for Agile Software Development Twelve Principles of Agile Software," 2001. Available: https://ai-learn.it/wp-content/uploads/2019/03/03_ManifestoofAgileSoftwareDevelopment-1.pdf
[17]. K. Schwaber, "SCRUM Development Process," *Business Object Design and Implementation*, vol. 1, no. 1, pp. 117–134, 1997, doi: https://doi.org/10.1007/978-1-4471-0947-1_11.
[18]. G. Ghantous and A. Gill, "Association for Information Systems AIS Electronic Library (AISeL) DevOps: Concepts, Practices, Tools, Benefits and Challenges," Jul. 2017. Available: https://opus.lib.uts.edu.au/bitstream/10453/130066/1/DevOps-%20Concepts%20Practices%20Tools%20Benefits%20and%20Challenges.pdf